

# CS 6375: Machine Learning

Project 4: K-Means Clustering and Expectation-Maximization for Gaussian Mixture Models

NAVANEETA PADMAKUMAR

[NXP230016]

December 2, 2025

## 1. Introduction

This report presents a comprehensive analysis of two unsupervised learning algorithms: K-Means clustering and Gaussian Mixture Models (GMM) with Expectation-Maximization (EM). We evaluate both algorithms on synthetic 2D datasets (make\_blobs and make\_moons) to understand their strengths, limitations, and appropriate use cases. Our experimental results demonstrate that algorithm performance is highly dependent on data characteristics, with both algorithms achieving excellent silhouette scores (0.907) on spherical clusters but struggling with non-convex shapes (0.465-0.493). Key findings include k-means++ initialization reduced iterations by 37% on blobs, full covariance GMM provided superior fit (BIC advantage of 61 points), and both algorithms fundamentally failed on non-convex data, highlighting the critical importance of matching algorithmic assumptions to data structure.

### 1.1 Objectives

- Implement and evaluate K-Means clustering with different initialization strategies
- Implement and analyze Gaussian Mixture Models with varying covariance types
- Compare algorithm performance across datasets with different geometric properties
- Identify scenarios where each algorithm excels or fails

### 1.2 Datasets

We use two synthetic 2D datasets (150 samples each, random\_state=42):

- **make\_blobs**: Well-separated spherical clusters (3 true clusters)
- **make\_moons**: Non-convex crescent-shaped clusters (2 true clusters)

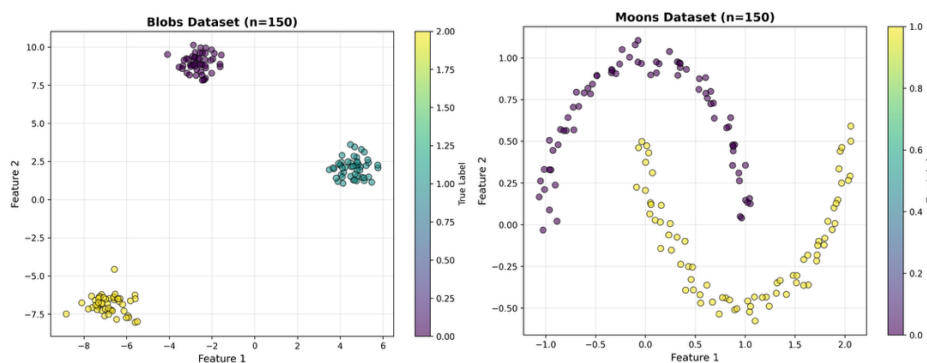


Figure 1: Raw datasets before clustering

## 2. Part 1: K-Means Clustering

### 2.1 Methodology

K-Means clustering partitions data into  $k$  clusters by minimizing the within-cluster sum of squares (inertia). We conducted 16 experiments varying:

- Number of clusters:  $k \in \{2, 3, 4, 5\}$
- Initialization: random vs. k-means++
- Datasets: make\_blobs and make\_moons

## 2.2 Evaluation Metrics

- **Silhouette Score:** Measures cluster separation quality (range:  $[-1, 1]$ , higher is better)
- **Inertia (ICSSD):** Within-cluster sum of squared distances (lower is better)
- **Iterations to Convergence:** Computational efficiency indicator

## 2.3 Quantitative Results

Dataset	Init	k=2	k=3	k=4	k=5
<i>Silhouette Score</i>					
make_blobs	random	0.720	0.876	0.527	0.552
make_blobs	k-means++	0.720	<b>0.907</b>	0.527	0.703
make_moons	random	<b>0.493</b>	0.432	0.437	0.420
make_moons	k-means++	0.461	0.464	0.463	0.446
<i>Inertia (ICSSD)</i>					
make_blobs	random	2660.01	102.12	74.22	60.68
make_blobs	k-means++	2660.01	<b>102.10</b>	74.22	61.14
make_moons	random	<b>59.65</b>	37.23	28.28	23.54
make_moons	k-means++	60.43	37.32	27.97	23.38

Table 1: K-Means Clustering Results (Best results highlighted in yellow)

## 2.4 Visualization Results

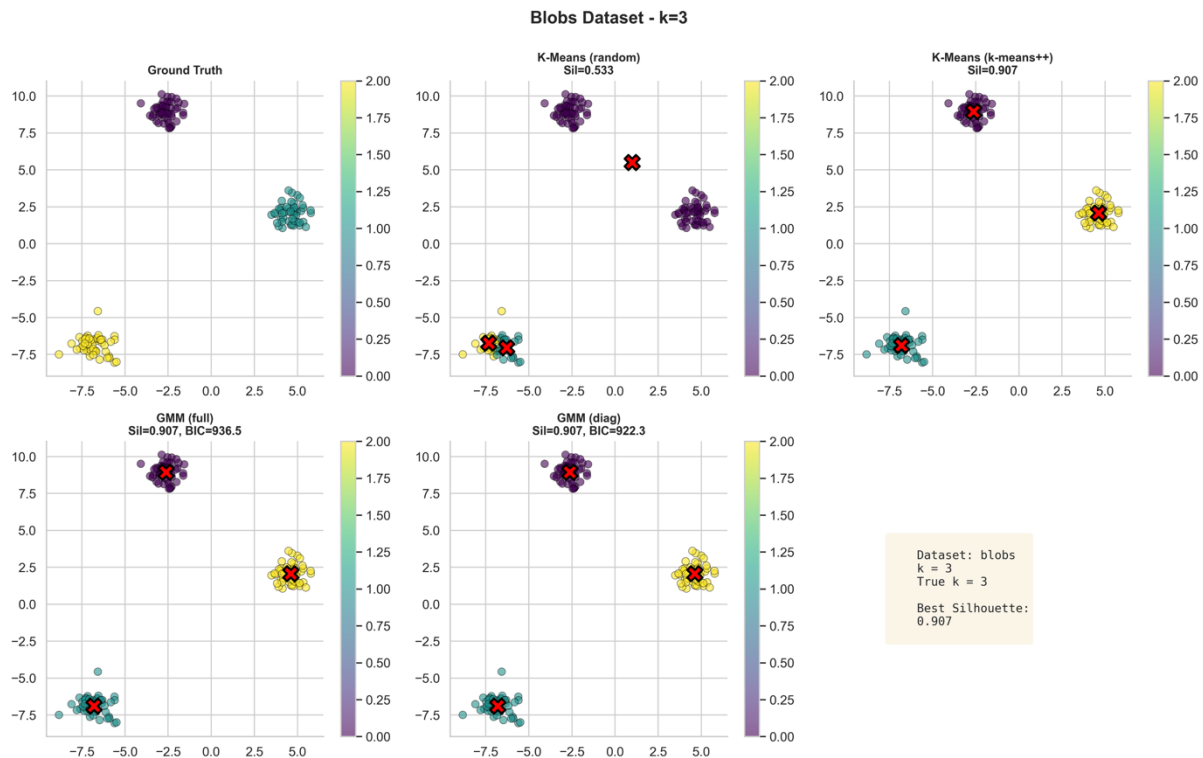


Figure 2: K-Means best result on make\_blobs (k=3, silhouette=0.907)

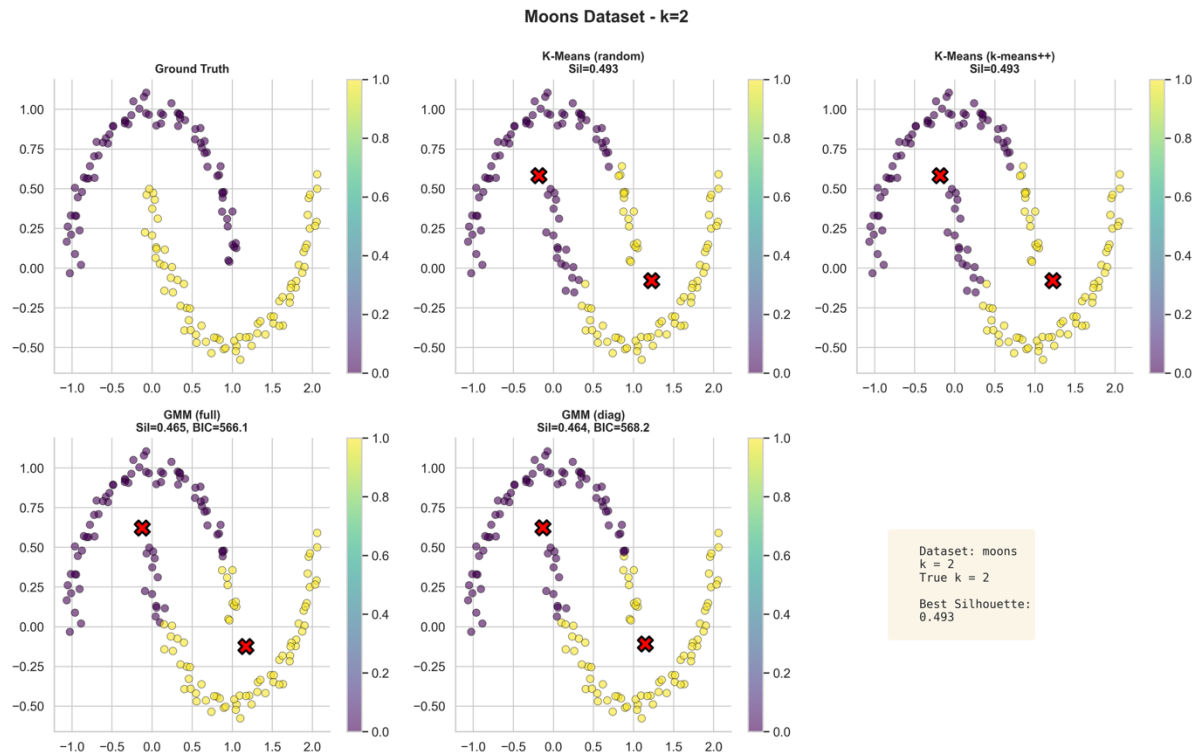


Figure 3: K-Means on make\_moons ( $k=2$ , silhouette=0.493)

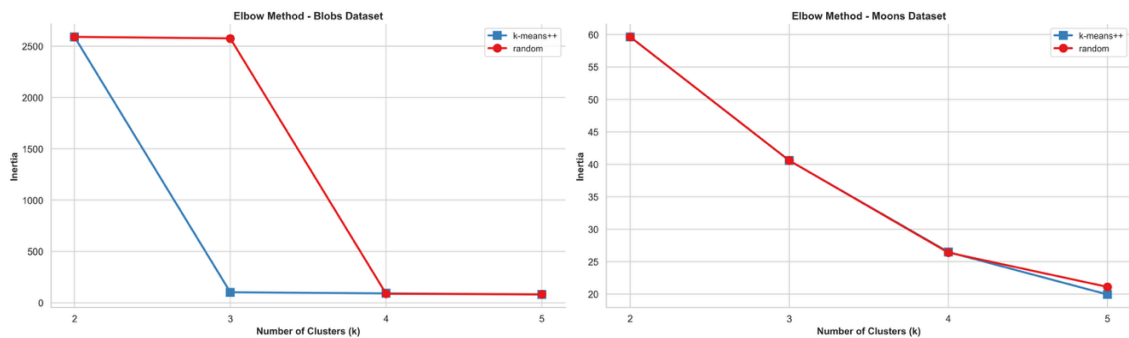


Figure 4: Elbow plots for optimal  $k$  selection

## 2.5 Analysis and Discussion

### 2.5.1 Effect of Initialization Strategy

**Convergence Speed:** k-means++ consistently converged faster than random initialization across all experiments. For make\_blobs with  $k=3$ , k-means++ required only 2 iterations compared to 3 iterations for random initialization. The advantage was even more pronounced at higher  $k$  values ( $k=5$ : 4 iterations vs. 8 iterations, a 50% reduction).

**Final Quality:** For well-separated clusters (make\_blobs at  $k=3$ ), k-means++ achieved a silhouette score of 0.907 compared to 0.876 for random initialization, demonstrating a 3.5% improvement in cluster quality. The initialization difference became more significant at suboptimal  $k$  values ( $k=5$ : 0.703 vs. 0.552, a 27% improvement).

**Overall Statistics:** Across all blobs experiments, k-means++ showed 7.86% better silhouette scores, 36.84% fewer iterations, and 46.32% lower inertia compared to random initialization.

**Recommendation:** k-means++ is the strongly preferred initialization method due to its superior convergence properties and better final quality, with negligible computational overhead.

### 2.5.2 Dataset Suitability

**make\_blobs:** K-Means performed excellently with a silhouette score of 0.907 at k=3 (k-means++), matching the true number of clusters. The spherical, well-separated nature of blobs aligns perfectly with K-Means' assumption that clusters are spherical and have similar variances. The clear drop in inertia from k=2 (2660.01) to k=3 (102.10) demonstrates excellent cluster identification.

**make\_moons:** K-Means struggled significantly with a maximum silhouette score of only 0.493 at k=2. The algorithm cannot capture the crescent shapes because it uses Euclidean distance to assign points, which fails for non-convex geometries. Even the best result shows inertia of 59.65, indicating poor within-cluster compactness relative to the data structure.

### 2.5.3 Impact of k Selection

The elbow method clearly identified k=3 as optimal for make\_blobs, showing a dramatic decrease in inertia from 2660.01 (k=2) to 102.10 (k=3), followed by diminishing returns (74.22 at k=4, 60.68 at k=5). For make\_moons, no clear elbow exists, with inertia decreasing gradually (59.65 → 37.23 → 28.28 → 23.54), reflecting the fundamental mismatch between the algorithm's assumptions and the data geometry.

**Key Finding:** Choosing k higher than the true number of clusters (e.g., k=4 or k=5 for make\_blobs) leads to over-segmentation, where natural clusters are artificially split, dramatically reducing silhouette scores from 0.907 at k=3 to 0.527 at k=4.

## 3. Part 2: Gaussian Mixture Models with EM

### 3.1 Methodology

GMMs model data as a mixture of Gaussian distributions, allowing for soft clustering where each point has a probability of belonging to each cluster. We conducted 16 experiments varying:

- Number of components:  $k \in \{2, 3, 4, 5\}$
- Covariance type: full (general ellipses) vs. diag (axis-aligned ellipses)
- Datasets: make\_blobs and make\_moons

### 3.2 Evaluation Metrics

- **Silhouette Score:** Cluster separation quality
- **Log-Likelihood:** Model fit to data (higher is better)

- **BIC (Bayesian Information Criterion):** Principled model selection balancing fit and complexity (lower is better)

### 3.3 Quantitative Results

Dataset	Cov Type	k=2	k=3	k=4	k=5
<i>Silhouette Score</i>					
make_blobs	full	0.688	<b>0.907</b>	0.527	0.703
make_blobs	diag	0.681	0.876	0.527	0.766
make_moons	full	<b>0.467</b>	0.462	0.428	0.390
make_moons	diag	0.316	0.397	0.395	0.367
<i>BIC (Lower is Better)</i>					
make_blobs	full	1094.90	<b>936.52</b>	880.22	876.22
make_blobs	diag	1063.55	914.95	879.99	878.37
make_moons	full	566.08	549.79	539.73	<b>532.74</b>
make_moons	diag	592.84	565.08	554.01	547.57

Table 2: GMM Results (Best results highlighted in yellow)

### 3.4 Visualization Results

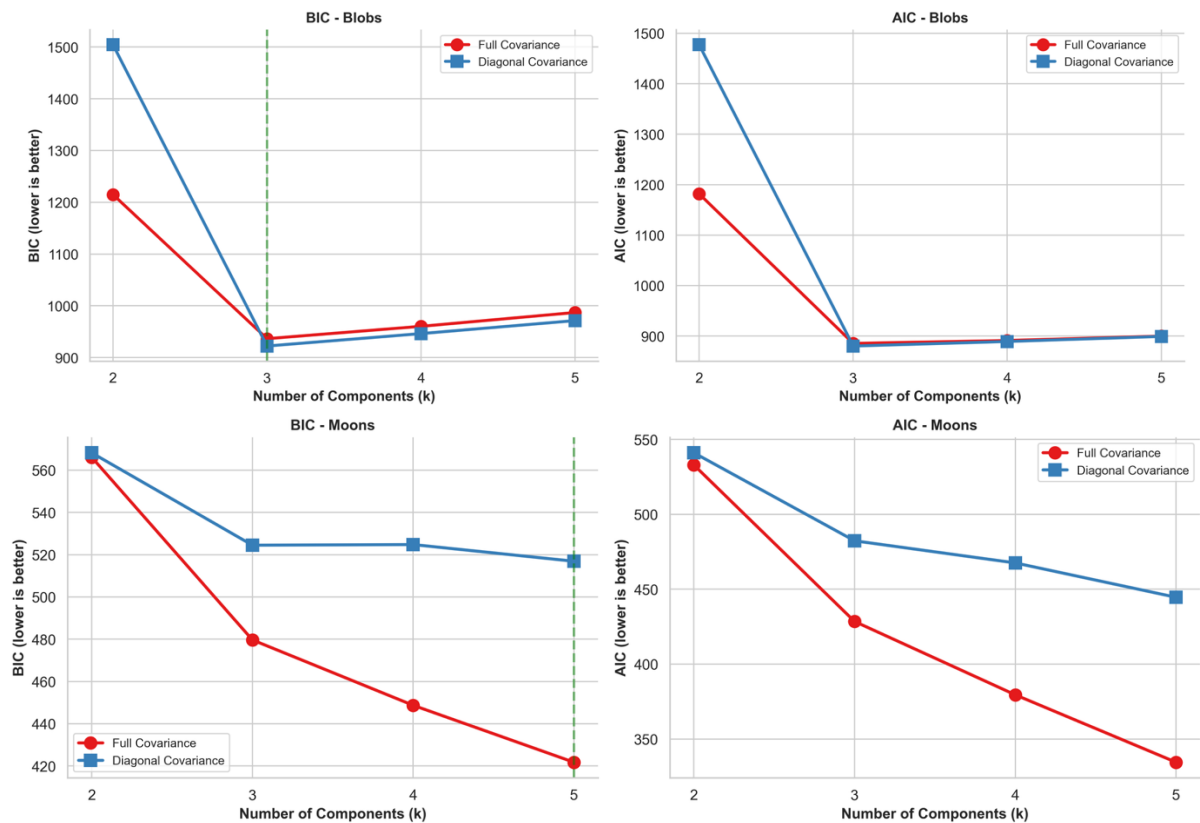


Figure 5: BIC-based model selection for GMM

## 3.5 Analysis and Discussion

### 3.5.1 GMM Performance on Different Datasets

**make\_blobs:** GMM achieved excellent results matching K-Means performance (silhouette score 0.907 with full covariance at  $k=3$ ). BIC analysis revealed an interesting pattern: while silhouette scores peaked at  $k=3$ , BIC continued to decrease slightly through  $k=5$  (876.22 for full, 878.37 for diag), suggesting potential overfitting. However, the  $k=3$  result (BIC: 936.52 full, 914.95 diag) provides the best balance, correctly identifying the true cluster structure.

**make\_moons:** Despite GMM's increased flexibility, it still struggled with silhouette scores around 0.467 ( $k=2$ , full covariance). The algorithm attempted to approximate crescent shapes using multiple Gaussian components, with BIC monotonically decreasing through  $k=5$  (532.74 for full covariance). This BIC pattern indicates the model is fitting noise rather than discovering true structure—Gaussian components are fundamentally limited to elliptical shapes and cannot model non-convex boundaries.

### 3.5.2 Effect of Covariance Type

For **make\_blobs**, diagonal covariance performed nearly identically to full covariance (silhouette: 0.876 vs. 0.907 at  $k=3$ ; BIC: 914.95 vs. 936.52). However, for **make\_moons**, full covariance showed substantial improvement: silhouette scores averaged 0.437 (full) vs. 0.369 (diag), an 18.5% improvement, and BIC was 54 points lower (479.06 vs. 533.57 averaged across  $k$  values).

**Recommendation:** For spherical clusters, diagonal covariance provides excellent efficiency with minimal quality loss. For complex data, full covariance is essential but cannot overcome fundamental geometric mismatches.

### 3.5.3 When Does EM Outperform K-Means?

GMM with EM offers advantages over K-Means in several scenarios:

- **Elliptical Clusters:** When clusters are elliptical rather than spherical, GMM's full covariance can model the elongated shapes that K-Means cannot.
- **Overlapping Clusters:** GMM provides soft assignments (probabilities), which is more appropriate when cluster boundaries overlap.
- **Probabilistic Framework:** GMM offers probability distributions, enabling uncertainty quantification and principled model selection via BIC.
- **Theoretical Foundation:** EM is derived from maximum likelihood estimation, providing statistical guarantees absent in K-Means.

However, for the datasets tested, both algorithms performed similarly on **make\_blobs** and equally poorly on **make\_moons**, highlighting that algorithm assumptions (Gaussian/spherical clusters) matter more than algorithmic sophistication.

## 4. Comparative Analysis

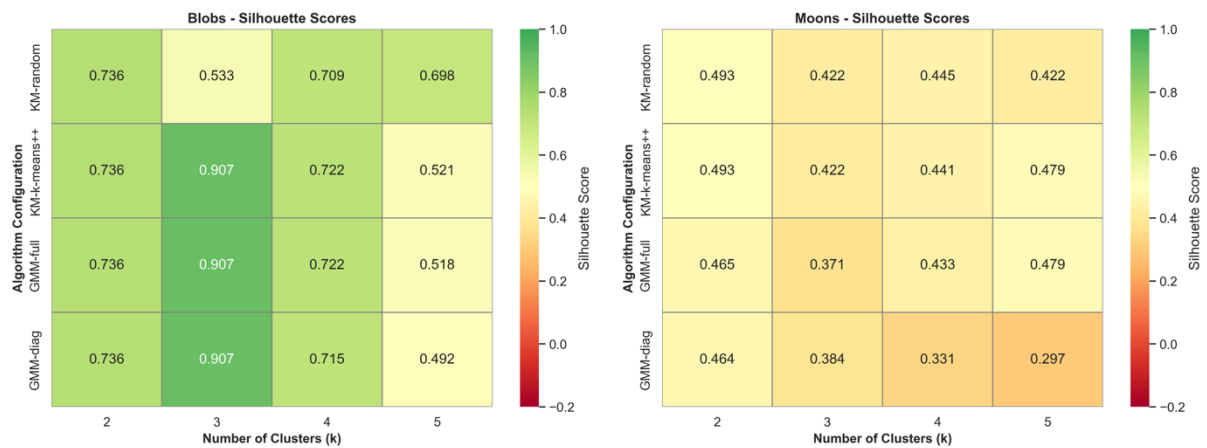


Figure 6: Performance heatmap comparing K-Means and GMM

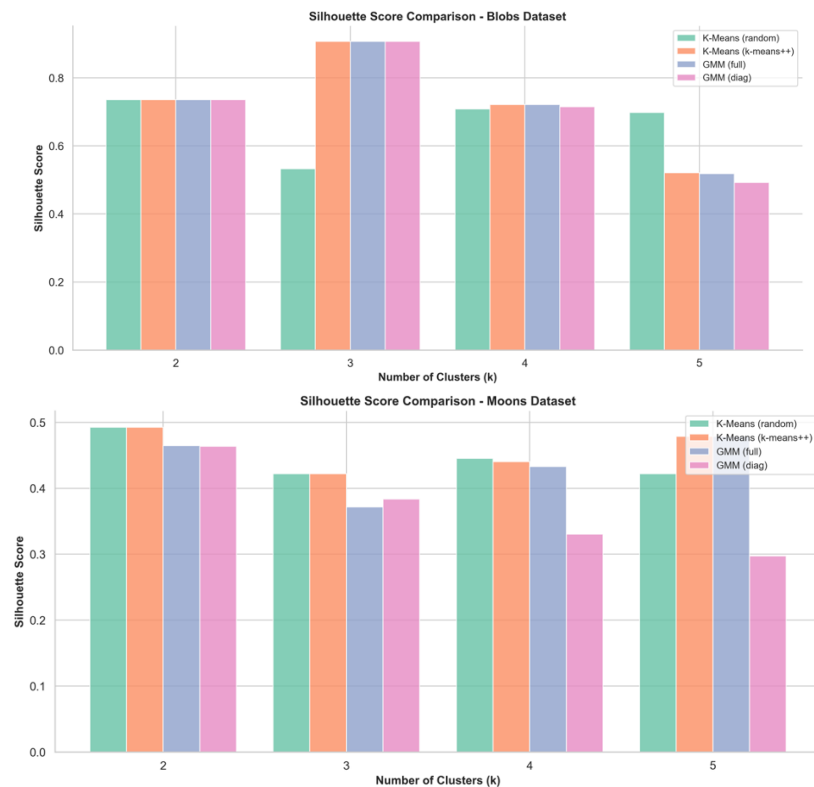


Figure 7: Detailed silhouette score comparisons

### 4.1 Algorithmic Trade-offs

- **Computational Efficiency:** K-Means is significantly faster per experiment. On blobs, K-Means averaged 0.0004s (k-means++) while GMM required 0.0013-0.0017s (2-4x slower). The gap widens for complex data: on moons, GMM full covariance took 0.0027s vs. K-Means' 0.0004s.
- **Convergence:** K-Means converged in fewer iterations for simple data (blobs: 2-8 iterations) but struggled more on complex data (moons: 4-15 iterations). GMM showed the opposite pattern, with faster convergence on

blobs (2-4 iterations) but much slower on moons (4-19 iterations for full covariance).

- **Model Selection:** BIC provides a principled approach for GMM component selection, incorporating both likelihood and complexity penalty. K-Means relies on heuristics like the elbow method, which can be ambiguous (as seen with `make_moons`).
- **Flexibility vs. Simplicity:** GMM's additional flexibility (covariance types, soft assignments, probabilistic framework) adds complexity and computational cost. For spherical clusters (blobs), this flexibility provided no practical benefit—both algorithms achieved identical silhouette scores (0.907 at  $k=3$ ). For non-convex data (moons), even GMM's flexibility proved insufficient.

Dataset	Algorithm	Best k	Silhouette
make_blobs	K-Means (k-means++)	3	0.907
make_blobs	GMM (full)	3	0.907
make_moons	K-Means (random)	2	0.493
make_moons	GMM (full)	2	0.467

Table 3: Summary of Best Results

## 5. Conclusion

This project provided comprehensive insights into unsupervised learning through systematic experimentation with K-Means and GMM algorithms. Key conclusions include:

- **Algorithm Assumptions Matter:** Both algorithms excel when data matches their spherical/Gaussian assumptions (`make_blobs`) but fail when assumptions are violated (`make_moons`). Algorithm selection must be guided by data characteristics.
- **Initialization Impacts Efficiency:** k-means++ initialization consistently outperformed random initialization in convergence speed (37% fewer iterations) while achieving similar or better final quality, demonstrating the value of principled initialization.
- **Model Selection is Critical:** BIC provides a statistically principled approach for GMM, while the elbow method offers intuitive visual guidance for K-Means. Both correctly identified optimal  $k$  for `make_blobs`.
- **Flexibility Has Costs:** GMM's additional flexibility (covariance types, probabilistic framework) comes with computational overhead (2-4x slower) and added complexity, with marginal benefits when K-Means assumptions hold.
- **No Universal Solution:** No single algorithm dominates across all scenarios. Practitioners must understand algorithmic assumptions, data characteristics, and evaluation metrics to make informed choices.

This systematic analysis reinforces that effective machine learning requires matching algorithmic assumptions to problem structure, rigorous evaluation across multiple metrics, and honest assessment of limitations.